Question 3 [40 points]

Inheritance and polymorphism are two intertwined essential concepts in OOP that allow hierarchical relationships among classes and enable working with different classes with the same interface. In Java, some of the mechanisms you can use to implement these concepts include ***extends***, ***super***, overriding methods, and overloading methods.

In this exam question, a base class `Product` is available for you. Do not modify Product.java. You must complete the following tasks.

Task 1 (10 points): Wine

Implement class `Wine` (in Wine.java), from scratch, that extends `Product`, with the following additional private variables:

```
private double baseVolume   //volume of this bottle of wine, in ml
private double actualVolume //actual volume of the wine, in ml
```

Then implement the constructor `Wine(double _basePrice, int _age, double _baseVolume)` that appropriately calls the superclass' constructor, and then set `baseVolume` and `actualVolume` to `_baseVolume`.

Wine generally becomes more expensive as it ages. Override `getPrice()` to reflect the following computation:

$$wine\_price = base\_price \cdot \frac{actual\_volume}{base\_volume} \cdot (1 + 0.05 \cdot age)$$

Furthermore, customers can complementarily taste (consume) the wine before purchasing, resulting in the actual volume of the wine being reduced. Therefore, implement an additional method:

`public void consume(double volume)`: Reduce the `actualVolume` by `volume`.

Running testWine() in `ProductTester` should give the following output:

```
@@ Wine ==> Original Prices:
Bottle 0: 500.00 baht
Bottle 1: 1600.00 baht
Bottle 2: 25197.20 baht
@@ Customers decided to taste some of the wine.
@@ Wine ==> After Prices:
Bottle 0: 450.00 baht
Bottle 1: 1600.00 baht
Bottle 2: 23349.41 baht
```

Task 2 (10 points): App

Implement class `App` (in App.java) that extends `Product`, and add the following variable:

```
private String appTitle; //Title of the app
```

Then implement the constructor `public App(double _basePrice, int _age, String _appTitle)` that appropriately calls the superclass' constructor, then set `appTitle` to `_appTitle`.

Since an app is software, its value decays as it ages. Override `getPrice()` to reflect the following computation:

$app\_price = base\_price \cdot 0.95^{age}$ `//base_price*(0.95^age)`

Hint: You may use Math.pow() to compute the exponential function.

Furthermore, implement the following additional methods:

`public String getAppTitle()` : Return the title of the app.

`public void update()`: Reset the age of the app to 0.

Running `testApp()` in `ProductTester` should give the following output:

```
@@ App ==> Original Prices:
Clubhouse: 84.88 baht
Angry Bird: 366.81 baht
Doctor Wins: 99.00 baht
@@ Let's update all the apps...
@@ App ==> Prices after updating.
Clubhouse: 99.00 baht
Angry Bird: 499.00 baht
Doctor Wins: 99.00 baht
```

Task 3 (20 points): SmartPhone

Implement class `SmartPhone` (in SmartPhone.java), by extending `Product`, and add the following class variables.

```
private String model = null;  //Model of this smartphone
private boolean used = false;
      //True if this is a used phone, false if this is a new phone.
private ArrayList<App> apps = null; //list of apps installed on this phone.
```

Then implement the constructor `public SmartPhone(double _basePrice, int _age, String _model, boolean _used)` that appropriately calls the superclass' constructor, and initializes `model`, `used`, and `apps`.

Since a smartphone is hardware with installed software, its price comprises the hardware price plus the installed apps' prices. Override `getPrice()` to reflect the following computation:

$$hardware\_price = base\_price \cdot 0.95^{age}$$

$$software\_price = \sum_{i \in apps} price\ of\ i$$

$$smartphone\_price = hardware\_price + software\_price$$

Then implement the following methods.

`public String getModel()`: Return the smartphone's model.

`public void install(App newApp)`: Add `newApp` to `apps`.

`public void install(App[] newApps)`: Add every app in `newApps` to `apps`.

`public void updateApp(String appTitle)`: Update the app whose title is `appTitle`.

`public void updateApp()`: Update every app installed on this phone.

Running testSmartPhone() in `ProductTester` should give the following output:

```
@@ SmartPhone prices before installing apps.
iPhone: 13995.33 baht
Oppo: 7347.24 baht
@@ SmartPhone prices after installing apps.
iPhone: 14569.42 baht
Oppo: 8665.63 baht
@@ SmartPhone prices after updating apps.
iPhone: 14618.07 baht
Oppo: 9163.24 baht
```

Files to submit: Wine.java, App.java, and SmartPhone.java